# YogaForm: Yoga Pose Correction AI Web Service Using Pose Estimation Model

Hye-In Kim[2020312578], Yu-Ha Roh-Choi[2019312564], and Jae-Hyun Song[2019311621]

SungKyunKwan University, Suwon 16419, Republic of Korea

**Abstract.** Yoga is an easy exercises to try at home alone, given that it can be done anytime, anywhere with just a single mat. But it is difficult to check your posture if you do yoga alone at home. To solve this problem, we propose a yoga pose correction AI web service "YogaForm". This service detects the user's posture using a posture estimation AI model and provides real-time feedback compared to the standard posture. Additionally, users can create a personalized routine by incorporating various offered postures. To implement this service, we used the React framework for the front-end, the Spring Boot framework for the back-end, and the Mediapipe AI framework for pose detection. Additionally, we implemented real-time data communication using RESTful API. By using this service, you can do yoga alone at home in the right posture with your own yoga routine. Therefore it will lead to improved user's home yoga experience.

**Keywords:** Pose Estimation · Pose Correction · Yoga.

## 1 Introduction

One of easy exercises to try at home alone is yoga. Because it can be done anytime, anywhere with just a single mat unlike other exercise that require certain equipment. The key to yoga is maintaining the correct posture. By sustaining proper alignment, various parts of the body can develop balance, enhancing flexibility and strength in muscles. Moreover, adopting the right posture aids in injury prevention and maximizes the effectiveness of yoga.

However, it is difficult to check your posture if you do yoga alone at home. Usually, there is no mirror room in home. even if there is, it's important not to look away during yoga, as doing so can increase the risk of injury. To solve this problem, we propose a yoga pose correction AI web service "YogaForm".

This service detects the user's posture using posture estimation AI model, and provides immediate feedback compared to the standard posture through TTS(Text-to-Speech) voice. It also offers many different postures so users can create personalized routine by themselves.

This web project combines front-end development using React and back-end development using Spring Boot. The hosting for the project is handled by an AWS server, and MySQL is employed as the database system. Mediapipe is

utilized as an AI framework for recognizing posture images. The project implements real-time data communication through RESTful API and polling mechanisms. RESTful API is utilized for efficient and swift data exchange between the front-end and back-end, while the polling mechanism provides real-time updates through a continuous connection with the server.

By using this service, you can do yoga alone at home in the right posture with your own yoga routine. Therefore it will lead to improved user's home yoga experience.

## 2 Design for the Proposed Service

### 2.1 Overall architecture

The overall system organization of the YogaForm is shown below.

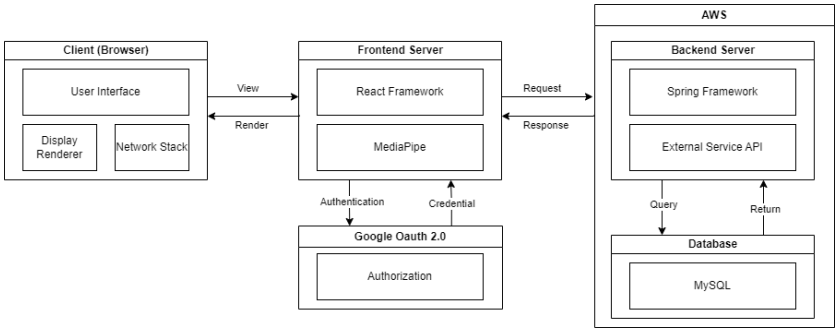


**Fig. 1.** Overall System Organization

The system is largely composed of three elements: user, front-end, and back-end. The user can access the system through a web browser, and a service request or response page to the request is rendered The front-end server handles all interactions with users. The service request sent by the user is parsed and delivered to the back-end server, and the page is rendered based on the response to the request and returned to the user. The front-end server is implemented using the React framework, and MediaPipe API is used to handle landmark detection. Also, Google Authentication is used to provide google log-in. The back-end server runs on AWS. It parses the request transmitted from the front-end server, performs the logic, and responds to the result. In the process, interactions with the database may occur, such as reading specific data or adding data. The back-end server is implemented using the Spring framework, and various libraries are also used to provide target services. MySQL is used as the database.

### 2.2 Reason for the design choice

**Using MediaPipe API** MediaPipe API is chosen in the system as a landmark detection model aside from other pose detection APIs for several reasons. First, MediaPipe is optimized for real-time performance, making it suitable for applications requiring quick and accurate pose estimation in live video streams. Its efficient design allows it to run smoothly even on devices with limited computational resources. Second, MediaPipe is versatile and supports multiple platforms, including mobile devices (iOS and Android), desktop, and the web. This flexibility makes our system accessible for a wide range of applications across different devices and operating systems.

**Sending landmarks to back-end** The unique part of our system architecture is that we apply MediaPipe in the front-end, not in the back-end. All the pose calculation and the error checking process takes place in the back-end. But, to do so, back-end should get the image from the front-end and then apply MeidaPipe to extract the landmark data. Here, the continuous image transferring process from front-end to back-end is required, and it can cause serious security issues. Therefore, we came out an idea to detect all the landmarks and send only the coordination to the back-end, and it might helps to avoid security problems.

### 2.3 Core technique

**Real Time Coaching with Polling Method** The main goal of this service is real-time yoga pose recognition and providing feedback. Upon detailed examination of this feature, the process involves extracting pose landmarks from the user's real-time webcam video using Mediapipe. Subsequently, the landmarks are calculated in real-time, and feedback is generated and delivered through voice. The service employs Polling techniques to implement real-time communication between the front-end and back-end, with bidirectional communication involving the transfer of landmark values from the front-end and voice feedback from the back-end.

In the front-end, Mediapipe is used to obtain the entire landmark values for each video frame, and these values are sent to the back-end every second. In the back-end, the received values are used to calculate the (x, y, z) coordinates of necessary landmarks, and the joint angles of arms, elbows, and knees are computed. The calculated angles are then compared to the angles stored from one second ago. If the current angles and the stored angles have a deviation within 20%, it is considered that the user maintains the same pose. Otherwise, if the deviation is beyond this threshold, it is interpreted as a correction in progress, and feedback is not generated. Feedback is only generated when the same pose is maintained.

Feedback requests from the front-end are sent immediately after confirming the transmission of landmark values, occurring at a 1-second interval. If no feedback content is generated by the back-end, a "no Content" value is sent to the front-end, preventing the reception of an audio file for playback. If feedback

content is generated, the value includes the audio content, and the back-end uses Google Cloud Text-To-Speech to create an audio file, which is then transmitted to the front-end. To prevent the transmission of feedback every second, the back-end sends a "no Content" response for 10 seconds after the last feedback transmission.

Upon receiving a value containing the audio file, the front-end requests the back-end to confirm success. During feedback generation, the average absolute percentage deviation of all joint angles is calculated. If all joints are within the acceptable deviation range, the front-end sends a value derived from subtracting the average error from 100. If there are significant errors in any joint, indicating a failure to achieve the pose, the front-end sends a value of -1. Based on this value, the front-end determines the success of the current pose and proceeds to the next pose accordingly.

**JWT Authentication** While using our service, users have the capability to create their own personalized yoga routines and review the results of their previous yoga sessions. Given the frequent need for user information exchange between the front-end and back-end, we identified potential security issues regarding the transmission of personal data. To address this concern, we opted to use JSON Web Tokens (JWT) to recognize users while preserving the confidentiality of their personal information.

JWTs are issued during the login process. The back-end generates a token and a refresh token with a validity period of 1 hours and 12 hours each and send tokens to the front-end as part of the login result. The front-end stores these tokens in the local storage. Subsequently, when requesting user information or performing various functions, the front-end includes these tokens in the header of the request sent to the back-end. The back-end, in turn, utilizes this token to identify the user by extracting their unique identifier, such as their email address, and then proceeds to retrieve and utilize the relevant data.

This implementation ensures that user information remains protected and confidential during communication between the front-end and back-end, allowing for secure user recognition through the utilization of JSON Web Tokens.

### 2.4 General challenges

**Webcam manipulation** To implement YogaCoach, one of the main functions in the system, the camera access is required on certain pages. Here, we encountered an issue wherein the camera access code from the preceding page persists and continues to run in the background upon routing to another page, leading to unintended runtime operation. Moreover, because there is no camera component in the other page, it causes an error that the component of the camera is undefined.

When playing video continuously, video data is provided through the stream object. To stop this, the following steps are necessary: 1. retrieve the currently used video stream from the webcam 2. stop all tracks associated with that stream

3. assign null to the video element to remove the connected stream. By applying these steps when routing to the other pages, we stopped the webcam video stream and cleaned up the used resources to resolve the issue.

**Loading Image files** Another main function of this system is Yoga Routine, displaying yoga poses and selecting poses to make user-own-routines. In YogaList page, the front-end requests the images of each yoga pose from the back-end and put them in the image array by iterating the pose name array which contains all the name of the yoga poses. Due to the asynchronous feature of React Framework, even if the map function iterates the pose name array in order, the image retrieval time from the back-end varies, resulting in discrepancy in the order of image array compared to the corresponding pose name array in certain proportions. Consequently, this results in the display of incongruent images alongside their respective pose names.

To address this challenge, we devised a strategy involving delayed execution upon detecting mismatched ordering. Specifically, we augmented the image retrieval function by incorporating the corresponding index parameter derived from the associated name array. When appending the images fetched from the back-end to the image array, the index is compared with the current length of the image array. If the index is bigger than the image array size, it means the preceding retrieval and appending process is not completed yet, thus we implemented a 300ms delay before appending the retrieved images.

## 3 Implementation

### 3.1 UI/UX Viewpoint

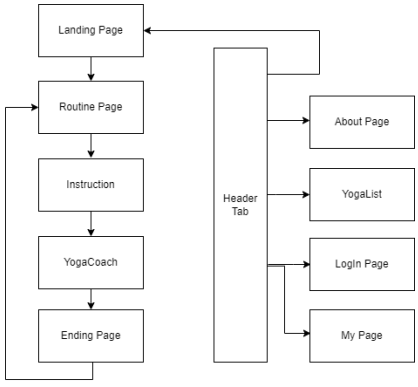


**Fig. 2.** Front-end System Architecture

### 3.2 Front-End

**React** As the front-end framework, React was chosen. Key specifications include:

- *React* version 18.2.0
- *npm* version 9.5.0

There are 9 pages as illustrated in Fig. 2. Front-end System Architecture.

*Landing Page* Landing Page is the initial screen that appears when you start the service. You can initiate a yoga routine by pressing the start button.

*Routine Page* When you press the start button on the landing page, it navigates to Routine Page where users can see a list of available yoga routines. If the user is logged out, only two routines, easy and hard, are displayed by default. If the user is logged in, the displayed routines include those created by the user. By selecting any routine, it will navigate to the instruction page.

*Instruction* Before starting a yoga routine, the system checks whether the user has an appropriate environment for using the service by verifying if the entire body is captured by the webcam. If landmarks, including the nose, both wrists, and both ankles(a total of five landmarks), are recognized, it is deemed that the user has a suitable environment for using the service, and it routes to the next page.

*Yoga Coach* The system retrieves poses one by one from the user's chosen routine and displays them. Alongside the standard pose image, the webcam screen is shown, allowing users to observe themselves. The system recognizes the user's poses and sends real-time landmark data to the server. The server then provides feedback, comparing it to the standard pose, in the form of an audio file. If the user maintains a pose with an error rate within 30%, it proceeds to the next pose. After completing all poses in the routine, it navigates to the next page.

*Ending Page* After completing all routines, Ending Page provides accuracy information for each pose within the routine and average.

*About Page* In the About page, the user can informed the main functions of YogaForm, how to access them, and the requirements for these functions.

*Yoga List* The Yoga List page can be navigated through the page header labeled "YOGA" and displays all the poses offered by the service. If the user is logged in, they can create a new routine by pressing the "Make a Routine" button. This button is not visible in the logged-out state. By selecting desired poses and providing a routine name, users can create their new routine.

*Login Page* YogaForm only accepts Google Authentication. The user can log in with Google Email. If the log-in succeeds, it will navigate to the landing page automatically. The log in lasts for 6 hours, thus user do not need to log in even if they reload our page.

*My Page* If the user logged in, the log-in button in the header becomes myPage button. Logged in user can access this page to see their routines and records. When the user select one of their routines, the average scores for each trial of the selected routine are expressed in a graph. Also, the poses inside the selected routine is displayed. In this page, user can log out by pressing the log-out button.

**MediaPipe** In Yoga Coach page, MediaPipe's Pose Detection API (`@mediapipe/pose`) is integrated to perform real-time pose estimation using a webcam feed. First, to detect landmarks from the camera, the `Camera` object from `@mediapipe/camera_utils` is utilized to stream the webcam's video frames to the MediaPipe Pose model. Later, `drawConnectors` and `drawLandmarks` from the `@mediapipe/drawing_utils` is used to draw the landmarks in the camera. To display the landmarks on the camera, we created a canvas which is the same size with the camera object and overlap it to mark each landmarks on the joint.

### 3.3 Back-End

**AWS** Utilizing AWS instances such as EC2 for hosting Spring Boot servers and RDS for database servers, a Spring Boot application was deployed alongside a MySQL database.

**Spring Boot** As the back-end framework, Spring Boot was chosen. Key specifications include:

- *Java* version 11
- *Spring Boot* version 2.7.15
- *Gradle* version 6.9.4

Employing the MVC pattern, the architecture consists of Controllers handling entry points where user requests are received, Services executing actual business logic and database operations via DAOs, and Models representing data entities. HTTP requests trigger controllers based on endpoints, ensuring a secure separation of concerns where controllers neither directly access the database nor execute critical logic.

Security configurations were implemented in the config package. CORS (Cross-Origin Resource Sharing) and CSRF (Cross-Site Request Forgery) functionalities were disabled to allow unrestricted communication between the front-end and back-end. Authentication is managed through JWT tokens, where endpoints with "secure/" in the API require a valid JWT token in the header. Other API calls can be made without authentication. To validate tokens, a JwtAuthenticationFilter was added.

To leverage JWT with Spring Security, the jjwt library version 0.11.5 was installed. In the JwtTokenProvider class, two tokens are generated based on the user's email: the primary token used to verify user information and a refreshToken, which automatically reissues a new token when the original token's validity period expires. These tokens are issued during login, allowing users to authenticate themselves using the token and decode it to extract the email of the user for whom the token was issued. This class is configured as a Bean object, enabling other components to utilize JWT seamlessly.

**MySQL**

– *MySQL* version 8.0.33.

Leveraged JPA (Java Persistence API) within the Spring framework. JPA represents a collection of interfaces that serve as the standard in the Java ecosystem for Object-Relational Mapping (ORM) technology. Implemented an entity structure, which mirrors the table structure stored in the database, and a DAO (Data Access Object) structure within the service layer to access the database server and execute SQL statements.

Established a total of four tables:

*User* Stores user information obtained during Google login, including the user's email.

*Pose* Stores information about collected poses, including pose names and standard values for each joint.

*Routine* Stores information about different routines, likely including details about the structure and composition of each routine.

*Record* Captures data related to each user's completion of a routine and the results derived from that session. This table may include information about the date and time of the record, as well as the specific routine and user associated with it.

By combining MySQL, JPA, and the defined table structures, you have established a robust foundation for effective data management, storage, and retrieval within your application.

## 4 Evaluation

**Empirical results** We conducted comprehensive in-house testing of our system, evaluating all functionalities across distinct user scenarios: 1) guest and 2) logged-in user. Our analysis, depicted in the use case diagram below, confirmed the successful execution of all identified use cases.

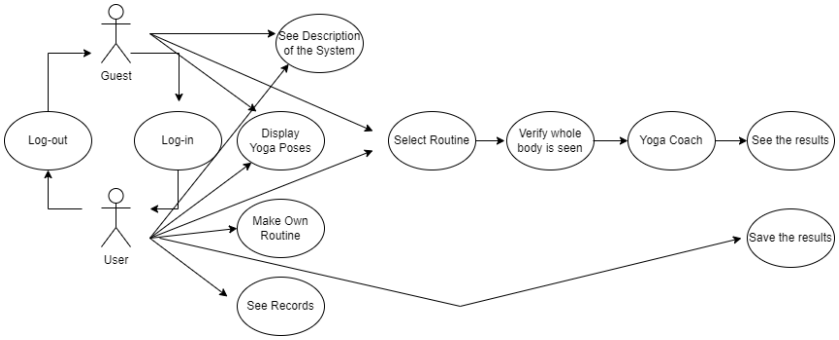


**Fig. 3.** Use case diagram

**Peer to peer feedback** Because our system is mainly concentrated on user experience, we should get the user feedback to evaluate our system. To do so, we go through the peer-to-peer feedback with the other team and responded their experiences. We get some positive feedback about great functionality of detecting landmarks meaning that the performance of MediaPipe API is trustworthy. We also received critical feedback concerning the inability to retrieve adjacent landmarks when they overlapped. This concern was mitigated by augmenting the landmark data to incorporate z-axis information, effectively resolving the issue.

## 5 Limitation and Discussions

### 5.1 Assumption

The service assumes that users are utilizing desktops or laptops, as it is a web-based service. It is expected that the camera on the user's device is functioning properly. While mobile usage is possible, the relatively small screens of mobile phones may limit the utility of the service. The assumption is that users will interact with the service individually, as errors may occur in the recognition process if multiple individuals are detected. It is also assumed that users are in a space where the entire body can be captured by the camera, and this condition is guided by instruction. Additionally, the service assumes that users are not wearing clothes that excessively obscure the silhouette of the body, as full-body recognition is necessary. All the information is provided on the about page.

### 5.2 Constraint and Limitation

The Mediapipe framework struggles to accurately recognize the neck joint in standard pose images, leading to the exclusion of poses where the neck overlaps with other body parts. Additionally, the limited number of available poses, currently offering only the standing pose, results in a lack of diversity in pose

options. Lastly, Due to the absence of yoga experts, difficulty levels for each yoga pose were arbitrarily set, and determining crucial considerations during the correction process is challenging.

### 5.3 Discussions

The main function of the project is to correct the user's posture by analyzing their joints. This can be applied not only to yoga but also to various exercises where proper posture is crucial, such as squats and golf. Additionally, by expanding the pose recognition feature of the project, a function that allows users to search for the names of yoga poses based on their movements can be developed. Discovering the name of a specific pose is only feasible when the user already knows it, and yoga poses, by their nature, often have lengthy and complex names. If users can identify the names of yoga poses solely by their movements, it would be beneficial for individuals practicing yoga on their own.

## 6 Related Work

**MixPose** MixPose[2] is a technology platform that specializes in AI-powered posture prediction and correction for various fitness and wellness activities, with a particular focus on yoga. MixPose leverages state-of-the-art AI algorithms and computer vision techniques to provide real-time feedback and guidance to users. The feedback provider is not AI, but the human instructor. This AI will estimate the pose and tell the overall accuracy of the pose, but can't provide specific correction guide or feedback by itself. Also it requires instructors who can stream their yoga class.

In contrast to MixPose, this service does not require a yoga expert during the pose correction process (although it may be needed in the development phase). YogaForm utilizes AI pose detection to assess and evaluate poses, providing real-time feedback based on accuracy. Additionally, it allows users to create routines by selecting poses they find necessary, and they can start their routines independently at their preferred times without relying on an instructor's yoga class.

## 7 Conclusion

In conclusion, the YogaForm project presents a novel approach to enhancing the home yoga experience through the integration of AI technology. By leveraging the capabilities of the MediaPipe AI framework for pose detection, the system provides real-time feedback on yoga poses, aiding users in correcting their postures during solo practice. The React and Spring Boot frameworks contribute to a seamless and responsive web service, while the use of JWT authentication ensures secure communication between the front-end and back-end.

The project successfully addresses the challenge of correcting yoga poses without the need for an expert instructor, making it accessible for users to practice yoga independently at home. The real-time coaching feature, coupled with personalized routine creation, enhances the overall user experience, promoting correct posture and contributing to the effectiveness of yoga sessions.

However, the system does have limitations, such as the dependency on proper webcam functionality and the exclusion of poses where the neck overlaps with other body parts. Additionally, the small number of available poses may limit diversity in pose options. To further enhance the system, future developments could focus on improving accuracy, especially in challenging poses. Another valuable addition could be enabling users to retrieve pose names based on their movements, enhancing the user experience.

Despite these limitations, YogaForm remains a valuable tool for users aiming to achieve correct postures and maximize the benefits of yoga exercises in the comfort of their homes.

## References

1. MediaPipe Homepage, https://developers.google.com/mediapipe. Last accessed 10 Dec 2023
2. MixPose Homepage, https://instructors.mixpose.com. Last accessed 10 Dec 2023
3. PocketYoga Homepage, https://www.pocketyoga.com/. Last accessed 10 Dec 2023
4. LNCS Homepage, https://kwonkai.tistory.com/141. Last accessed 10 Dec 2023
5. GoogleOAuth Homepage, https://developers.google.com/identity/protocols/oauth2?hl=ko. Last accessed 10 Dec 2023
6. GoogleTTS Homepage, https://cloud.google.com/text-to-speech?hl=ko. Last accessed 10 Dec 2023